

**EXTENDING A STANDARD-BASED REMOTE FILE ACCESS  
PROTOCOL AND MAINTAINING COMPATIBILITY WITH  
A STANDARD PROTOCOL STACK**

**Inventor(s)**

Christos Karamanolis  
1657 Belleville Way, Apt. J  
Sunnyvale, CA 94087

Zheng Zhang  
3520 Casabella Court  
San Jose, CA 95148

Mallik Mahalingam  
620 Park View Drive, #105  
Santa Clara CA 95054

**Assignee**

Hewlett Packard Company

1                   **EXTENDING A STANDARD-BASED REMOTE FILE ACCESS**  
2                   **PROTOCOL AND MAINTAINING COMPATIBILITY WITH**  
3                   **A STANDARD PROTOCOL STACK**

4                   **FIELD OF THE INVENTION**

5                   The present invention generally relates to standards-based protocols, and more  
6                   particularly to an arrangement and method for extending a standard-based protocol  
7                   while maintaining compatibility with standard-based protocol stack.

8

9                   **BACKGROUND**

10                  The Network File System (NFS) is a standard network protocol that makes files  
11                  stored on a file server accessible to any computer (client) on a network. The NFS client  
12                  protocol provides access to files that reside on the remote server, in the same way as a  
13                  client's local file system provides access to files that are stored on a local disk. NFS is  
14                  transparent to client applications and exposes a traditional file system interface.

15                  The NFS client protocol makes Remote Procedure Calls (RPCs) to the remote  
16                  server in order to implement access to the files. The interface is implemented by the  
17                  NFS protocol using RPCs. The RPC layer also provides methods for data  
18                  representation that are independent of the host architecture. For example, when a  
19                  client application needs to write to a file that resides in a remote server, it calls a  
20                  generic write() system call. The NFS client protocol implements the write() call by  
21                  invoking a request to the remote server via the RPC layer. RPC marshals the data and  
22                  parameters of the write() call into a message and encodes the latter in an architecture  
23                  independent format. The message is sent to the remote host via the TCP/IP protocol  
24                  stack. It is delivered up by the TCP/IP stack of the server to the RPC layer, where it is  
25                  unpacked and passed to the NFS server protocol. The NFS server protocol implements  
26                  the write() operation by accessing the server's local file system. A reply is sent back to  
27                  the client through the NFS, RPC and TCP/IP protocol layers.

28                  The RPC layer uses standard external data representation (XDR) for the  
29                  information exchanged between different computer architectures. NFS on a file server  
30                  provides a transparent, stateless access to shared data.

31                  Improving the performance of a standard protocol such as NFS will often  
32                  involve deviating from or extending the standard. In order to implement such  
33                  improvements, both the client-side NFS protocol and the server-side NFS protocol

1 may require enhancements. This creates hurdles for both vendors and potential  
2 customers of improved systems. While a vendor of an improved NFS-based file server  
3 could wait and hope that its extensions to the standard are adopted, the time and effort  
4 involved in changing standards along with rapidly changing market conditions, may  
5 render this approach unworkable.

6 A system and method that address the aforementioned problems, as well as  
7 other related problems, are therefore desirable.  
8

### 9 **SUMMARY OF THE INVENTION**

10 In various embodiments, an apparatus and method are provided for extending a  
11 standard-based protocol while maintaining compatibility with the standard protocol  
12 stack. Network file system (NFS) remote procedure calls (RPCs) that are submitted by  
13 an NFS client application are intercepted in a manner that is transparent to the client  
14 operating system. The intercepted NFS-RPCs are sent to a file interface card. The file  
15 interface card includes a processor that executes code that implements the standard  
16 NFS-RPC protocol, along with extensions to the standard NFS-RPC protocol. Non-  
17 NFS RPCs are sent to a native (OS) communication protocol stack, and from the OS  
18 protocol stack the non-NFS RPCs are submitted to a conventional network interface  
19 card (NIC). The processor on the file interface card also provides TCP/IP, UDP and  
20 other protocol processing. Non-NFS RPCs are forwarded to a native protocol stack and  
21 are sent over a network via a conventional network interface card (NIC).

22 It will be appreciated that various other embodiments are set forth in the  
23 Detailed Description and Claims which follow.  
24

### 25 **BRIEF DESCRIPTION OF THE DRAWINGS**

26 Various aspects and advantages of the invention will become apparent upon  
27 review of the following detailed description and upon reference to the drawings in  
28 which:

29 FIG. 1 is a functional block diagram of a system that includes an extended file  
30 interface card in accordance with one embodiment of the invention;

31 FIG. 2A illustrates the relationship between an RPC interceptor layer and other  
32 client software layers in accordance with an example embodiment of the invention;

33 FIG. 2B illustrates the relationship between a packet filter layer and other client  
34 software layers in accordance with another embodiment of the invention;

1 FIG. 3 is a functional block diagram of the extended file interface card  
2 according to one embodiment of the invention;

3 FIG. 4 is a flowchart of a process that implements the RPC interceptor layer of  
4 FIG. 2A;

5 FIG. 5 is a flowchart of a process that implements the packet filter layer of FIG.  
6 2B;

7 FIG. 6 is a flowchart of a process that implements an NFS client proxy  
8 according to one embodiment of the invention.

9

## 10 **DETAILED DESCRIPTION**

11 In various embodiments of the invention, an NFS client proxy interacts with  
12 NFS servers having extensions to the standard NFS protocol, and other system entities  
13 such as storage area network (SAN) devices, name servers and other server systems.  
14 The NFS client proxy is implemented on the extended file interface card (EFIC), which  
15 also runs a complete network stack, including the TCP/UDP stack and IP layer. By  
16 implementing the proprietary portion of the protocol on the interface card, performance  
17 is enhanced and the proprietary interactions are hidden from the standard NFS software  
18 on the client system. With the present invention, extensions that seek to improve  
19 performance, scalability, and fault-tolerance of distributed file systems, for example  
20 can be implemented in a manner that minimizes adaptation of the client system.

21 FIG. 1 is a functional block diagram of a system that includes an extended file  
22 interface card in accordance with one embodiment of the invention. System 100  
23 includes a client data processing system 102 that is coupled to server systems 104 and  
24 106 via a standard network 108. Server system 104 is a system that hosts an extended  
25 version of the standard NFS protocol, and block 106 represents one or more server  
26 systems with which the NFS client proxy 110 interacts. For example, the other types of  
27 servers include standard NFS servers, storage area network (SAN) devices, meta-data  
28 servers, name servers, or database servers.

29 Client system 102 includes a conventional processor 112 that is coupled to the  
30 extended file interface card (EFIC) 114 and a conventional network interface card 116  
31 via the host I/O bus 118 (e.g., PCI bus). The NFS client proxy 110 is implemented on  
32 the EFIC 114, along with the standard TCP/UDP stack 118, IP layer 120, and physical  
33 layer 121. The NFS client proxy implements functional extensions added to the  
34 standard NFS protocol. Network interface card 116 provides a standard interface for

1 client system 102 to the network 108. For example, network interface card 116 may be  
2 any one of a number of commercially available cards for connecting a Unix, Linux,  
3 NT, or Windows based machine to a network.

4 Processor 112 hosts software elements NFS client 122, interceptor module 124,  
5 EFIC interface 126, and native protocol stack 128. NFS client is an example client  
6 application that makes NFS-RPC calls. Interceptor module 124 is a software module  
7 that works in conjunction with EFIC 114. The function of interceptor module 124 is to  
8 intercept RPC calls, and depending on the particular procedure referenced, direct the  
9 call to either EFIC interface 126 or to native protocol stack 128 for processing. NFS-  
10 specific RPC calls are directed to EFIC interface 126, and other RPC calls are directed  
11 to native protocol stack 128. Two alternative implementations of interceptor module  
12 are described in FIGs. 2A and 2B, and in FIGs. 4 and 5. EFIC interface 126 forwards  
13 the extended NFS-RPC calls to EFIC 114, and network card interface forwards the  
14 other RPC calls to network interface card 116.

15 The upper layers of the standard NFS client protocol are implemented in the  
16 operating system kernel (not shown) of the client system. The upper layers of the NFS  
17 client protocol implement the generic file system operations, such as read(), write(),  
18 open(), etc., and in some cases generate messages that are sent via the RPC layer. This  
19 functionality is implemented in the OS kernel and is not affected by interceptor module  
20 124 or by the NFS client proxy 110.

21 The lower layers of the standard NFS client protocol, along with any extensions  
22 to the standard protocol, are implemented in the NFS client proxy element 110 on the  
23 EFIC 114. The lower layers include functionality that is related to the construction and  
24 transmission of NFS-specific RPC messages. For example, the lower layers control  
25 how the RPC messages (for NFS-read(), NFS-write(), etc.) are packed, encoded, and  
26 where the messages are sent over the TCP/IP stack. In different embodiments, the  
27 servers to which the RPC messages are sent and the interactions with the servers are  
28 extended in non-standard ways in the EFIC. For example, one RPC message submitted  
29 by the NFS client protocol may be interpreted and result in several other RPC and non-  
30 RPC messages being generated and sent to entities on the network such as meta-data  
31 servers, name servers, etc.

32 Server system 104 is an example data processing system that hosts extended,  
33 non-standard NFS protocol software 132. The extended NFS protocol software 132  
34 interacts with NFS client proxy 110 in accordance with extended NFS protocols. The

1 server system 104 also includes conventional TCP/IP stack 134 and physical layer 136  
2 for interfacing with network 108.

3 By implementing the NFS-RPC services in the EFIC 114 instead of the  
4 operating system, legacy systems can be upgraded to take advantage of extensions to  
5 NFS by installation of the EFIC and accompanying interceptor module 124. The  
6 present invention also removes the burden of having to rebuild and reinstall the  
7 operating system kernel to upgrade the client system 102. Once the EFIC 114 is  
8 installed in the client system 102, further enhancements can be made to the client's  
9 NFS interface by downloading new software to the EFIC, rather than having to upgrade  
10 the client's operating system.

11 FIG. 2A illustrates the relationship between an RPC interceptor layer and other  
12 client software layers in accordance with an example embodiment of the invention.  
13 The software elements 122, 126, 152, 154 and 156 are hosted by processor 112. RPC  
14 interceptor 152 is implemented as a software module that is plugged into the head of a  
15 stream queue (not shown) between the NFS and RPC stacks in the OS kernel. Some  
16 operating systems, for example, HP-UX, provide support for streams. Streams is a  
17 framework provided by operating systems to facilitate a modular configuration of  
18 system software, in terms of layers, where each layer conforms to a standardized  
19 interface. Streams provide a generic mechanism for passing data and control  
20 information between software layers. The RPC interceptor examines message headers  
21 of messages in the stream queue for NFS-specific RPCs. The NFS-specific RPCs are  
22 then presented to EFIC interface 126, which transmits the messages to the EFIC 114. It  
23 will be appreciated that with the RPC interceptor approach, all processing below NFS is  
24 off-loaded to the EFIC 114 for both standard and non-standard NFS RPCs.

25 For messages referencing other RPCs (non-NFS related), RPC interceptor 152  
26 lets the message flow through to the native RPC layer 154. RPC layer 154  
27 communicates with native TCP/IP stack 156 to send and receive messages over  
28 network 108 via network interface card 116.

29 FIG. 2B illustrates the relationship between a packet filter layer and other client  
30 software layers in accordance with another embodiment of the invention. Packet filter  
31 162 is a software module that intercepts packets below the RPC layer 154 in the OS  
32 kernel. The RPC layer receives an out-bound message from the system software  
33 module above it, for example, the NFS client protocol, and marshals the data and  
34 control parameters of that message into a contiguous sequence of bytes. The marshaled

1 message is encoded using XDR functions in a format that is independent of host  
2 architecture and is submitted to the TCP/IP (or UDP/IP) protocol stack for  
3 transmission. In addition, the RPC protocol is responsible for retransmission of the  
4 message if a reply is not received in a selected period of time, and is responsible for  
5 other functionality related to management of system resources , congestion control, etc.  
6 For in-bound messages delivered from the TCP/IP (or UDP/IP) stack, RPC decodes the  
7 message, un-marshals the contents, and delivers the message to the appropriate system  
8 module above (e.g. NFS). The packet filter 162 opens the RPC packet and examines  
9 the function code. If the function code references an NFS-specific RPC, the packet is  
10 directed to EFIC interface 126. Non-NFS RPCs are forwarded to the native TCP/IP  
11 stack 156.

12 The embodiment of FIG. 2A having the RPC interceptor offers the advantage  
13 over the embodiment of FIG. 2B having the packet filter of reduced computation costs.  
14 The packet filter embodiment requires opening a messages after it has been marshaled  
15 and encoded by the RPC layer, which is computationally expensive. In contrast the  
16 RPC interceptor examines the message prior to marshaling and encoding, thereby  
17 saving host processor cycles. It will be appreciated, however, that the RPC interceptor  
18 requires an operating system that supports streams.

19 FIG. 3 is a functional block diagram of the extended file interface card  
20 according to one embodiment of the invention. EFIC 114 includes a packet parser 202,  
21 a processor 204, a memory 206, and a system interface 208. In operation, packet parser  
22 202 interrogates a received information packet. The packet header is stored in memory  
23 206 for further processing, and depending on the state of the protocol, the data portion  
24 of the packet is either also stored in memory 206 or sent to system interface 208.  
25 System interface 208 sends the data to the host processor for further processing.

26 Processor 204 is configured to interrogate the packet header information that is  
27 stored in memory 206 in accordance with one or more network protocol handling  
28 operations (e.g., firewall filtering, load balancing, and TPC/IP or RTP protocol  
29 handling operations). In addition, processor 204 is configured for protocol interactions  
30 as defined by the extensions to the NFS-RPC protocol.

31 System interface 208 includes, in one embodiment, a direct memory access  
32 (DMA) controller that is configured to arbitrate between EFIC 114, host memory (not  
33 shown, and other devices that perform DAM transfers over host I/O bus 118. Further  
34 details regarding an example implementation of EFIC 114 can be found in the

1 application/patent entitled, "PROCESSING NETWORK PACKETS", by Russell et al.,  
2 filed on August 11, 2000, having application/patent number 09/630,033, and assigned  
3 to the assignee of the present invention. The contents of the application/patent are  
4 incorporated herein by reference.

5 FIG. 4 is a flowchart of a process that implements the RPC interceptor layer of  
6 FIG. 2A. The RPC interceptor intercepts messages at the head of the stream queue  
7 between the OS kernel NFS and RPC components. While the process of FIG. 4  
8 illustrates the operations performed on only one message, those skilled in the art will  
9 appreciate that all the messages in the stream are intercepted and processed as  
10 described in the flowchart. At step 302, a message is intercepted from the stream. If  
11 the message is an NFS-specific RPC message, decision step 304 directs the process to  
12 step 306.

13 At step 306, the message is sent to the NFS client proxy on the EFIC 114. The  
14 client proxy performs on behalf of the host processor 112 the client-related NFS  
15 operations, as well as any non-standard extensions to the NFS operations. Messages  
16 returned from the client proxy are returned to the client application at step 308 to  
17 complete the process.

18 For non-NFS related messages, decision step 304 directs the process to step  
19 310. Non-NFS messages are forwarded to the native RPC over TCP/IP stack for  
20 standard processing. At step 312, messages returned from the native RPC stack are  
21 returned to the NFS client.

22 FIG. 5 is a flowchart of a process that implements the packet filter layer of FIG.  
23 2B. The NFS-RPC packet filter intercepts RPC packets below the NFS-RPC elements  
24 of the operating system kernel. While the process of FIG. 5 illustrates the operations  
25 performed on only one packet, those skilled in the art will appreciate that all the packets  
26 from the RPC layer are intercepted and processed as described in the flowchart. At step  
27 352, a packet is intercepted from the RPC layer. If the function code in the RPC packet  
28 NFS-specific, decision step 354 directs the process to step 356.

29 At step 356, the message is sent to the NFS client proxy on the EFIC 114. The  
30 client proxy performs on behalf of the host processor 112 the client-related NFS  
31 operations, as well as any non-standard extensions to the NFS operations. Messages  
32 returned from the client proxy are returned to the client application at step 358 to  
33 complete the process.

1       For non-NFS related RPC packets, decision step 354 directs the process to step  
2 360. Non-NFS RPC packets are forwarded to the native TCP/IP stack for standard  
3 processing. At step 362, messages returned from the native TCP/IP stack are returned  
4 to the NFS client.

5       FIG. 6 is a flowchart of a process that implements an NFS client proxy  
6 according to one embodiment of the invention. The NFS client proxy 110 processes  
7 NFS-specific RPC requests on behalf of the client operating system kernel, thereby off-  
8 loading RPC processing from the host processor and supporting extensions to the  
9 standard NFS protocols.

10       At step 402, the NFS client proxy receives a packet from the interceptor module  
11 124. The function code is obtained from the NFS-RPC packet at step 404. The  
12 function code dictates the protocol by which the NFS client proxy will interact with the  
13 addressed server system. At step 406, the NFS client proxy performs the protocol  
14 processing dictated by the function code in the packet. For example, for standard NFS  
15 functions (functions that have not been extended), the NFS client proxy interacts with  
16 the server in a manner consistent with the standard. For extended NFS functions (non-  
17 standard), the NFS client proxy interacts with the server in the manner defined by the  
18 extended function.

19       When the NFS client proxy reaches a state in the interaction with the server  
20 system where the protocol specifies, the data or response is returned to the host  
21 processor, as shown by step 408. The client proxy repeats this general process for each  
22 NFS-RPC packet received from the host processor.

23       The present invention is believed to be applicable to a variety of computer  
24 system architectures and with a variety of operating systems. Other aspects and  
25 embodiments of the present invention will be apparent to those skilled in the art from  
26 consideration of the specification and practice of the invention disclosed herein. It is  
27 intended that the specification and illustrated embodiments be considered as examples  
28 only, with a true scope and spirit of the invention being indicated by the following  
29 claims.

30

31